Md. Alif Hassan
ID : 20-42870-1

Activation Function - Report

Activation functions in neural networks introduce non-linearity and enable neural networks to learn complex functions. In this report, I will cover six different activation functions and their math formulas with pros and cons.

1. **Step Function :** This is a simple activation function that matches any input greater than or equal to from zero to one and all inputs less than zero to zero.The mathematical formula for the step function is as follows:

   f(x) = 1, if x >= 0
   f(x) = 0, if x < 0

   Pros :

- Simple and computationally efficient
- Can be used as a binary classifier

  Cons :

- Not differentiable at x = 0, making it unsuitable for use in backpropagation-based learning algorithms.
- Gradient updates are constant and not dependent on the input, making it difficult to optimize.

2. **Sigmoid Function :** This function takes any real value as input and outputs values in the range of 0 to 1. The larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to 0.0. The mathematical formula for the sigmoid function is as follows:

   f(x) = 1 / (1 + e^(-x))

   Pros :

   - Smooth and differentiable
   - Can be used to model probabilities, making it useful for binary classification problems

- Well-suited for use in shallow neural networks with few hidden layers

Cons :

- Prone to the vanishing gradient problem, making it difficult to train deep neural networks.
- Outputs are not zero-centered, which can slow down convergence during training.

3. **Tanh Function :** Tanh function is very similar to the sigmoid/logistic activation function, and even has the same S-shape with the difference in output range of -1 to 1. In Tanh, the larger the input (more positive), the closer the output value will be to 1.0, whereas the smaller the input (more negative), the closer the output will be to -1.0. The mathematical formula for the tanh function is as follows:

f(x) = (e^x - e^(-x)) / (e^x + e^(-x))

Pros :

- Smooth and differentiable.
- Outputs are zero-centered, making it more effective for training deep neural networks than the sigmoid function.
- Well-suited for use in shallow and deep neural networks.

Cons :

- Prone to the vanishing gradient problem, making it difficult to train very deep neural networks.

4. **ReLu Function :** ReLU stands for Rectified Linear Unit.
Although it gives an impression of a linear function, ReLU has a derivative function and allows for backpropagation while simultaneously making it computationally efficient. The main catch here is that the ReLU function does not activate all the neurons at the same time. The neurons will only be deactivated if the output of the linear transformation is less than 0**.**

f(x) = max (0, x)

Pros :

- Computationally efficient

- Can accelerate convergence during training
- Does not suffer from the vanishing gradient problem

Cons :

- Not differentiable at x = 0, which can cause issues during backpropagation.
- Prone to the "dying ReLU" problem, where a large portion of the network can become non-responsive and stop learning.

5. **PRelu Function :** Parametric ReLU is another variant of ReLU that aims to solve the problem of gradient's becoming zero for the left half of the axis.
   This function provides the slope of the negative part of the function as an argument a. By performing backpropagation, the most appropriate value of a is learnt.

   f(x) = max (0, x) + alpha * min (0, x)

   Pros :

   - Can address the "dying ReLU" problem.
   - Can learn the slope of the negative part of the function during training, leading to improved performance.

   Cons :

   - More computationally expensive than the ReLU function.
   - May not always improve performance over the ReLU function

6. **ELU Function :** Exponential Linear Unit, or ELU for short, is also a variant of ReLU that modifies the slope of the negative part of the function.
   ELU uses a log curve to define the negative values unlike the leaky ReLU and Parametric ReLU functions with a straight line.

   f(x) = x, if x > 0
   f(x) = alpha * (exp(x) - 1), if x <= 0

   where alpha is a hyperparameter that controls the magnitude of the negative slope.

   Pros :

   - Addresses the "dying ReLU" problem by preventing the gradient from becoming zero or negative for negative inputs, which can speed up training.

- Smooth and differentiable, which helps with optimization and gradient-based learning .
- Outputs are zero-centered, which can improve learning performance.

Cons :

- More computationally expensive than the ReLU function and other variations like the PReLU function.
- Additional hyperparameters to tune (alpha).
- Not as widely used or well-known as other activation functions, so there may be less community support and fewer resources available for implementation and optimization.

The End